

Requested Patent: JP7200317A
Title: OPERATION RIGHT MANAGEMENT EQUIPMENT ;
Abstracted Patent: JP7200317 ;
Publication Date: 1995-08-04 ;
Inventor(s): TAKAHASHI TOSHINARI; others: 04 ;
Applicant(s): TOSHIBA CORP ;
Application Number: JP19930349335 19931228 ;
Priority Number(s): ;
IPC Classification: G06F9/46; G06F1/00; G06F12/14 ;
Equivalents: ;

ABSTRACT:

PURPOSE: To attain flexible operation right management for threads by providing a means storing revision right information and a means referencing the stored revision right information before execution of revision of operation right information and verifying whether or not the revision is permitted to the equipment.

CONSTITUTION: The equipment is provided with a means storing revision right information representing whether or not revision of operation right information is permitted based on a thread or the user being a subject of the revision or a memory area or a program in which the subject of revision is in existence and a means referencing the stored revision right information before execution of the revision of the operation right information and verifying whether or not the revision is permitted. That is, an operation right list storage section 7 stores a table relating to thread protection and storing a fact of the right executing the operation to each thread and used for an operation right discrimination section 12. Thus, not only the management of the operation right to the thread but also the revision right is flexibly expressed and managed by the combination of various conditions.

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平7-200317

(43) 公開日 平成7年(1995)8月4日

(51) IntCl. ⁶	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 9/46	3 4 0 B	7629-5B		
1/00	3 7 0 E			
12/14	3 1 0 K			

審査請求 未請求 請求項の数 3 F D (全 14 頁)

(21) 出願番号 特願平5-349335

(22) 出願日 平成5年(1993)12月28日

(71) 出願人 000003078

株式会社東芝

神奈川県川崎市幸区堀川町72番地

(72) 発明者 高橋 俊成

神奈川県川崎市幸区小向東芝町1番地 株

式会社東芝研究開発センター内

(72) 発明者 岡本 利夫

神奈川県川崎市幸区小向東芝町1番地 株

式会社東芝研究開発センター内

(72) 発明者 福本 淳

神奈川県川崎市幸区小向東芝町1番地 株

式会社東芝研究開発センター内

(74) 代理人 弁理士 鈴江 武彦

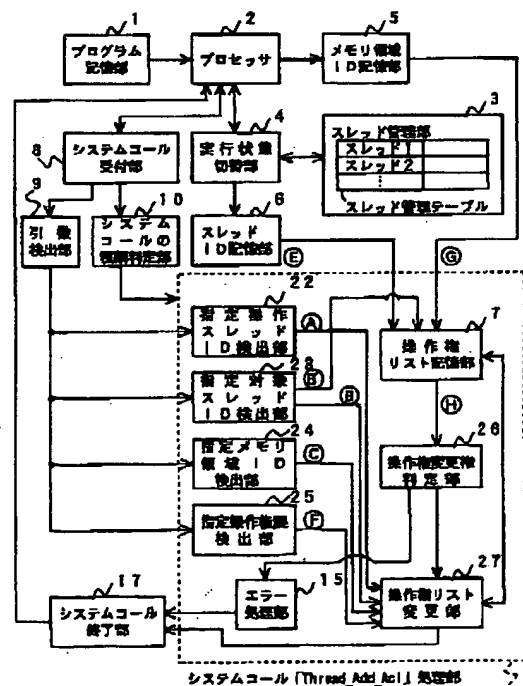
最終頁に続く

(54) 【発明の名称】 操作権管理装置

(57) 【要約】

【目的】 スレッドの操作権管理の柔軟化を目的とする。

【構成】 本発明は、複数の並行して実行されるスレッドによりプログラムを実行する手段と、各スレッドに対する操作を、この操作を行う主体となるスレッドまたはユーザ、もしくはこの操作を行う主体が存在するメモリ領域またはプログラムの少なくともいずれかに基づいて、許可するか否かを指示する操作権情報を記憶する手段とを備える操作権管理装置において、前記操作権情報の変更を、この変更を行う主体となるスレッドまたは、ユーザもしくはこの変更を行う主体が存在するメモリ領域またはプログラムの少なくともいずれかに基づいて、許可するか否かを示す変更権情報を記憶する手段と、前記操作権情報の変更を実行する前に、記憶された前記変更権情報を参照して、この変更が許可されるか否かを検証する手段とを具備したことを特徴とする。



【特許請求の範囲】

【請求項1】複数の並行して実行されるスレッドによりプログラムを実行する手段と、

各スレッドに対する操作を、この操作を行う主体となるスレッドまたはユーザ、もしくはこの操作を行う主体が存在するメモリ領域またはプログラムの少なくともいずれかに基づいて、許可するか否かを指示する操作権情報を記憶する手段とを備える操作権管理装置において、前記操作権情報の変更を、この変更を行う主体となるスレッドまたは、ユーザもしくはこの変更を行う主体が存在するメモリ領域またはプログラムの少なくともいずれかに基づいて、許可するか否かを指示する変更権情報を記憶する手段と、

前記操作権情報の変更を実行する前に、記憶された前記変更権情報を参照して、この変更が許可されるか否かを検証する手段とを具備したことを特徴とする操作権管理装置。

【請求項2】実メモリまたは仮想メモリを複数のメモリ領域の集合として管理する手段と、

各メモリ領域に対する操作を、この操作を行う主体となるスレッドまたはユーザ、もしくはこの操作を行う主体が存在するメモリ領域またはプログラムの少なくともいずれかに基づいて、許可するか否かを指示する操作権情報を記憶する手段とを備える操作権管理装置において、前記操作権情報の変更を、この変更を行う主体となるスレッドまたは、ユーザもしくはこの変更を行う主体が存在するメモリ領域またはプログラムの少なくともいずれかに基づいて、許可するか否かを指示する変更権情報を記憶する手段と、

前記操作権情報の変更を実行する前に、記憶された前記変更権情報を参照して、この変更が許可されるか否かを検証する手段とを具備したことを特徴とする操作権管理装置。

【請求項3】前記変更権情報を前記操作権情報に付随されて記憶することを特徴とする請求項1または2に記載の操作権管理装置。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、計算機システム中でのメモリ領域またはスレッドへの操作権を管理する操作権管理装置に関するものである。

【0002】

【従来の技術】近年マイクロプロセッサの性能向上により複数の小型のコンピュータをネットワークでつないで使用することが通常のこととなり、ネットワーク結合された計算機やユーザー同士での協調作業が可能となり、操作権の管理を厳密に行う必要が出てきた。

【0003】またオブジェクト指向技術の進歩やサーバ・クライアント・モデルのような新しいパラダイムの出現により、従来の記憶装置やファイルを単位とした操作

権管理では十分な性能が得られなくなってきた。

【0004】こういった問題を解決するために、単一仮想記憶などの技術により、メモリやCD-ROMやハードディスク・ドライブのような記憶装置、さらにはディスプレイやキーボードといったあらゆる計算機資源を同一の概念で管理するオペレーティング・システムの設計が試みられている。中でも、各種の資源を同一の仮想空間上に配置する単一仮想記憶の技術は特徴的なものである。さらに、この単一仮想記憶をいくつかのブロックに分割し、ブロック単位で操作権を管理する方法が提案されている。この管理単位を「メモリ領域」と呼び、あらゆる記憶装置の操作権管理の単位として用いられる。

【0005】一方、オペレーティング・システムはマイクロ・カーネル技術の定着により、プログラムの実行を管理するスレッドという単位を使って管理するのが一般的になりつつある。すなわち、プログラム実行の制御対象としてプロセスまたはスレッドと呼ばれる制御単位が用いられる。個々のスレッドは独立にプロセッサのレジスタ値やスタックなどを持ち、オペレーティング・システムによりプロセッサを割り当てられた間だけプログラムを実行し、一定時間後またはハードウェア割り込みなどが発生したときにオペレーティング・システムはスレッドに対するプロセッサの割り当てを解除して他のスレッドへプロセッサを割り当てる。個々のスレッドは独立したレジスタ値を持っているため、仮想的に並列に動いているプロセッサであるとみなすことができる。

【0006】このようなオペレーティングシステムでは、メモリ領域とスレッドという二つの概念を単位として計算機資源を構成し、そこへの操作権を管理する情報もACL (Access Control List) という形で各メモリ領域および各スレッドに付随させて管理することが行われる。

【0007】例えば、スレッドの場合を例に説明すると、スレッド自体を操作したいとき、すなわちスレッドの実行を一時停止したり再開したりしたいときやスレッドを強制的に消滅させたいとき、またはスレッドの情報(レジスタ値やスタック等の内部状態)を調べたいときのため、オペレーティングシステムはスレッド操作命令群を備えており、あるスレッドから別のスレッドを操作することが可能である。このようなスレッド操作命令群は、ユーザプログラムからシステムコールとしてオペレーティング・システムに指示し、利用できるようになっている。

【0008】このようなスレッド操作命令があるときに、どのスレッドからでもスレッド操作命令を発行できるようにすると、不当な操作もできることになってしまうので保護の機能が必要になる。たとえばスレッドを強制的に消滅させるようなことは特定の権限を持つスレッドだけから行えるようにすることが必要になる。

【0009】さらに、仮想空間が複数あり、仮想空間ご

とに1つのプログラムやデータが配置されており、スレッドもその空間内しか動きまわれないようなMac h (参考文献:「分散オペレーティング・システム」前川守他編、共立出版)をはじめとするオペレーティング・システムの場合、同じ空間内に存在するスレッドの保護を同一とし、スレッドの保護をメモリ空間の保護に依存する方法をとっている。

【0010】しかし、それらのスレッドを区別して保護することはできない。また、単一仮想空間のシステムのように、1つのアドレス空間内にすべてのプログラムが配置され、複数のスレッドが存在する場合、スレッドの保護はメモリ保護だけではなくオペレーティングシステムによる権限の設定が必要になる。

【0011】この問題のひとつの解決法は、スレッドの所有者を決めて、同一の所有者のスレッド間ではスレッドの操作を許す方式であり、Unixオペレーティングシステムのsignalはこれに相当する。しかしこれでは同一の所有者のスレッド間の保護の柔軟性に欠け、さらに、操作権限を他の所有者のスレッドに渡すことができない。また、複数の所有者のスレッドからのスレッド操作を可能にしたりできないなど、柔軟にスレッドの操作ができるようなスレッド保護機能が不足していた。

【0012】また、一方ではデータ・ファイルへの操作権の多彩な設定ができるように、ファイルやディレクトリに対するACLに、情報の追加権、削除権、ACLの変更権などのさまざまな情報を付加して柔軟性を高めようとするDCE (参考文献:OSF DCE技術解説、ソフトリサーチセンター) のような試みもあるが、やはり複数の所有者にメモリ領域のACLの管理を行わせるような柔軟な保護機能は不足している。

【0013】

【発明が解決しようとする課題】以上述べてきたことに基づいて従来の操作権管理装置の主要な課題をまとめると次のようになる。

【0014】(1) 従来、スレッドに対する操作を、この操作を行う主体や、この操作を行う主体が存在するメモリ領域などに基づいて、許可するか否かを指示する方法は試みられている。しかし、スレッドに対するこの操作権情報を変更する権利については、例えば、そのスレッドの所有者には認める、あるいはルートと呼ばれる特別のユーザにのみ認める、といった単純な管理しか行うことができなかった。

【0015】(2) また、従来、実メモリ、仮想メモリ、データ・ファイルなどのメモリ領域に対する操作を行う主体や、この操作を行う主体が存在するメモリ領域などに基づいて、許可するか否かを指示する方法は試みられている。しかし、メモリ領域に対するこの操作権情報を変更する権利については、例えば、そのメモリ領域の所有者には認める、あるいはルートと呼ばれる特別のユーザにのみ認める、といった単純な管理しか行うこと

ができなかった。

【0016】(3) さらに、スレッドやメモリに対する操作権を柔軟に記述し、かつ、同様に変更権をも柔軟に記述すると、柔軟な記述ができるようになる反面、スレッドやメモリに対する保護のためのデータ量が多くなり、その管理が複雑になるという欠点を持つという問題がある。

【0017】このように、従来の操作権管理装置では、メモリ領域単位での操作権限の設定ができなかったり、操作権限の設定の面で柔軟性に欠けたり、または、同一の仮想アドレス空間を共有しているスレッド間では保護が無かったか、または制約が強かった。

【0018】本発明は、上記課題を考慮してなされたものであり、別々の仮想空間にいるスレッド間ではもとより、同一仮想アドレス空間を共有しているスレッド間でのスレッド操作命令あるいは個々のメモリ領域に対して、同一の方法で柔軟性があり、かつ十分な保護を行う操作権管理装置を提供することを目的とする。

【0019】

【課題を解決するための手段】上記目的を達成するために、本発明(請求項1)では、複数の並行して実行されるスレッドによりプログラムを実行する手段と、各スレッドに対する操作を、この操作を行う主体となるスレッドまたはユーザ、もしくはこの操作を行う主体が存在するメモリ領域またはプログラムの少なくともいずれかに基づいて、許可するか否かを指示する操作権情報を記憶する手段とを備える操作権管理装置において、前記操作権情報の変更を、この変更を行う主体となるスレッドまたは、ユーザもしくはこの変更を行う主体が存在するメモリ領域またはプログラムの少なくともいずれかに基づいて、許可するか否かを示す変更権情報を記憶する手段と、前記操作権情報の変更を実行する前に、記憶された前記変更権情報を参照して、この変更が許可されるか否かを検証する手段とを具備したことを特徴とする。

【0020】また、本発明(請求項2)では、実メモリまたは仮想メモリを複数のメモリ領域の集合として管理する手段と、各メモリ領域に対する操作を、この操作を行う主体となるスレッドまたはユーザ、もしくはこの操作を行う主体が存在するメモリ領域またはプログラムの少なくともいずれかに基づいて、許可するか否かを指示する操作権情報を記憶する手段とを備える操作権管理装置において、前記操作権情報の変更を、この変更を行う主体となるスレッドまたは、ユーザもしくはこの変更を行う主体が存在するメモリ領域またはプログラムの少なくともいずれかに基づいて、許可するか否かを示す変更権情報を記憶する手段と、前記操作権情報の変更を実行する前に、記憶された前記変更権情報を参照して、この変更が許可されるか否かを検証する手段とを具備したことを特徴とする。

【0021】また、望ましくは、上記発明(請求項1)

において、スレッド作成時には、あらかじめ定められたデフォルトの変更権情報が設定されることを特徴とする。

【0022】また、望ましくは、上記発明（請求項2）において、メモリ領域作成時には、あらかじめ定められたデフォルトの変更権情報が設定されることを特徴とする。

【0023】さらに、上記各操作権管理装置において、前記操作権情報を記憶する手段の情報を変更することが不可能になるような、前記変更権情報を記憶する手段の情報の変更を認めないようにしても良い。

【0024】また、本発明（請求項3）では、上記発明（請求項1または2）において、前記変更権情報を前記操作権情報に付随されて記憶することを特徴とする。

【0025】さらに、上記各操作権管理装置において、前記操作権情報を記憶する手段の情報を変更することのできる複数の条件を記憶する操作権管理権限記憶手段と、前記変更権情報を記憶する手段の情報に変更を加えようとする際に該操作権管理権限記憶手段の情報によりこの変更を行う権限があるか否かを検証する手段と、この権限があると検証された場合に前記変更を許可する手段とをさらに設けても良い。また、この場合、前記変更権情報を記憶する手段の情報を変更することが不可能になるような、前記操作権管理権限記憶手段の情報の変更を認めないようにしても良い。

【0026】

【作用】本発明（請求項1）によれば、スレッドへの操作権の管理のみならず、その操作権の管理の権限である変更権についても、さまざまな条件の組み合わせによって柔軟に表現、管理することができるようになる。

【0027】この機構を用いれば、スレッドの操作権に関して、特定のスレッドにのみその変更を許可したり、またそれらの複数の条件を組み合わせることで表現したりすることができる。

【0028】特に、近年多く利用されるようになったグループウェアと呼ばれる計算機の協調作業アプリケーションを作成する際には、スレッドの共有や、排他制御、他のプログラムからの保護、などのきめ細かな制御を容易に記述することが必要になるが、本発明を用いればこの記述を柔軟に行うことができる。

【0029】また、プログラムの誤りや、システム・コールの不正な呼び出しなどによって、スレッドへの操作権が異常に書き換えられてしまうことを未然に防ぐことができるので、スレッドの共有を行いつつもなおスレッドの動作に関する信頼性が飛躍的に向上する。

【0030】また、本発明（請求項2）によれば、メモリ領域への操作権の管理のみならず、その操作権の管理の権限である変更権についても、さまざまな条件の組み合わせによって柔軟に表現、管理することができるようになる。

【0031】この機構を用いれば、メモリ領域への操作権に関して、特定のスレッドにのみその変更を強化したり、特定のメモリ領域に存在するスレッドにのみその変更を許可したり、またそれら複数の条件を組み合わせることで表現したりすることができる。

【0032】特に、近年利用が試みられている単一仮想記憶と呼ばれる、複数のアプリケーションが同一の仮想空間上で動作するオペレーティング・システムにおいては、メモリ領域の共有や、排他制御、他のプログラムからの保護、などのきめ細かな制御を記述することが必要となるが、本発明を用いればこの記述を柔軟に行うことができる。

【0033】また、プログラムの誤りや、システム・コールの不正な呼び出しなどによって、メモリ領域への操作権が異常に書き換えられてしまうことを未然に防ぐことができるので、メモリ領域に書かれたデータが不用意に書き換えられてしまうことがおこらず、メモリ領域の共有を行いつつもなおメモリ領域に書かれたデータに対する信頼性が飛躍的に向上する。

【0034】さらに、本発明（請求項3）によれば、スレッドやメモリに対する操作権情報と、その操作権情報の変更を行う変更権情報とは、一般に共通点が多く、データ構造も似通ったものになることが多い。本発明においては、変更権情報を操作権情報に付随させて記憶することにより、共通のデータをなるべく一つのものとして管理するため、データ量を少なくすることができ、変更権および操作権を検証するための手続きも共通化する。

【0035】また、スレッドやメモリに対する操作権や変更権などの管理は、一般にメモリ管理ユニットと呼ばれるハードウェアを用いることによって、実行性能を向上させることができる。そのような場合に、操作権および変更権それぞれに別のハードウェアを用意することは、計算機の構成を複雑にする可能性がある。しかし、本発明によって、操作権と変更権を同一に管理することにより、必要なハードウェアを単純化することができる。

【0036】

【実施例】以下、図面を参照して本発明の実施例について説明する。

【0037】本発明に係る操作権管理装置は、オペレーティング・システム（OS）と呼ばれる計算機システム全体の実行制御を行うものの構成要素の一部である。OSは計算機資源であるメモリ、プロセッサ、周辺機器を管理し、また、ユーザプログラムとよばれるユーザが指示した実務上の処理を行うプログラムの実行を制御する。一つの装置上には複数のユーザプログラムがのっており、OSがそれらのプログラム間の制御も行う。

【0038】ユーザプログラムは、実行上に必要となる物理メモリなどの資源を確保したり、周辺機器にデータを入出力したり、他のユーザプログラムにデータを転送

したり、処理を依頼したり、実行を制御するためにOSに対して指示を行うことが必要となる。このOSに対する指示をシステムコールと呼ばれる特別な手段で行う。本実施例では、システムコールのうち、操作権を考慮するユーザプログラムの制御に関して説明する。

【0039】操作権については、スレッドの管理とメモリ領域の管理がある。これらメモリ領域の管理とスレッドの管理については同一の部分が多いので、同様の機構を重複して説明することは避け、主にスレッドに対する操作権管理を例として、同一の部分には同一の用語を用いて説明する。

【0040】なお、本実施例では、メモリの「操作」とは、アクセス（読み書き実行）のことを表すものとする。

【0041】最初に、本実施例の特徴を概略的に述べる。本実施例では、個々のメモリ領域またはスレッドに対して、どのような条件で操作を許可するかという情報を登録したリストを設け、このリストにより許可されている場合に限り、操作を許可するとともに、この情報自体を変更することのできる権限を、前記リストと同一の形式、類似の形式あるいは共通の形式で管理する。また、特定の条件の時にのみこの変更を許す機構を付加する。また、この変更をする権限の変更についても同一の形式、類似の形式あるいは共通の形式で管理する。

【0042】この結果、本実施例は、不当な操作命令やプログラムの誤りからメモリ領域やスレッドを保護し、またこのような誤った操作命令を検出することによってプログラムの誤りの検出を容易とするために、メモリ領域またはスレッドごとに設けた操作の可否を示す情報を変更する権利自体もそのメモリ領域またはスレッドが持つので、操作権の管理のみならず、操作権の管理の権限すらも同一の形式、類似の形式あるいは共通の形式で管理し、削除し、追加し、または変更することを柔軟に行うことができるようになる、という作用効果を奏するものである。

【0043】また、この操作権の管理の権限は、メモリ領域やスレッドの所有者とは全く別の機構より、通常のアクセス制御リストと同様の形式で複数持つので、操作権管理の権限を複数のスレッドで共有したり、自分の作ったメモリ領域やスレッドの管理権を他のスレッドに譲渡したりするなどの柔軟な操作権管理が可能となる。

【0044】（第1の実施例）以下、本発明の第1の実施例について説明する。

【0045】図1は、本実施例の全体構成図である。プログラム記憶部1には、計算機で処理する命令が記述されたユーザプログラムや処理する対象のデータ、計算機で処理途中の結果を一時的に記憶するスタックなどのデータがメモリ空間上に蓄えられている。メモリ空間は、複数の領域（メモリ領域）に区切られて管理され、それぞれの領域には、お互いを区別するための識別子がつい

ており、メモリ領域ごとにプログラムやデータやスタックを配置している。

【0046】さらに、プログラム記憶部1に蓄えられているプログラムを解釈実行し、データを処理する実行主体であるプロセッサ2と、プロセッサ内の状態を管理するスレッド管理部3、プロセッサの実行状態の切り替え処理を行う実行状態切り替え部4がある。

【0047】プロセッサ2内には、命令の実行途中に一時的に利用するレジスタ類、現在実行している命令のプログラム記憶部での位置を示すためのプログラムカウンタ（PC）、スタックデータの先頭位置を示すスタックポインタ（SP）などが存在し、これら一連のデータを入れ替えることで、複数の別々のユーザプログラムの処理を時分割で並行して進めることが可能となる。

【0048】スレッドとは、上記プロセッサ内で使用する一連のデータのことを呼び、これを複数組用意し、このスレッドを切り替えてプロセッサにセットし実際に実行させることにより、時分割処理が可能となる。

【0049】メモリ領域ID記憶部5は、現在実行中のプログラムがプログラム記憶部1のどのメモリ領域に存在しているかを、そのメモリ領域のIDとして記憶しているものである。これは、プロセッサ2内のPCの値を利用し、現在実行中のメモリ領域IDを求めて、記憶している。このメモリ領域IDは、現在実行中のプログラムIDとみなすこともできる。

【0050】スレッド管理部3の中にはスレッド管理テーブルと呼ぶ表があり、プロセッサ2にセットするレジスタの情報やその他の情報がスレッドごとに管理されている。スレッドには、お互いを区別するために識別子（スレッドID）がついている。

【0051】実行状態切替部4は、現在プロセッサ2上で実行しているスレッド（例えばスレッドID=1）を中断し、スレッド（スレッドID=1）の状態をスレッド管理テーブルの所定の位置（スレッドID=1の格納領域）に退避し、別のスレッド（例えばスレッドID=2）を起動すべく、スレッド管理テーブルのスレッド（スレッドID=2）の情報をプロセッサ2に再セットし、スレッド（スレッドID=2）の実行を再開させる。また、実行状態切替部4には、さらに、どのスレッドをセットし退避するかを決定するスレッドスケジューリング機構（図示せず）と、いつスケジューリングするか指示するタイマ部（図示せず）が存在する。

【0052】スレッドID記憶部6は、スレッドIDを記憶するためのものである。

【0053】操作権リスト記憶部7は、スレッド保護に関するテーブルを保持し、スレッドごとにスレッドに対する操作を実行する権限がどうなっているかを記憶するもので、操作権限判定部12で使用される。本実施例はスレッドの操作権管理を例として説明しているが、メモリ領域の操作権管理の場合には、この操作権リスト記憶

部7はメモリ領域の保護に関するテーブルを保持し、スレッドごとにメモリ領域に対する操作を実行する権限がどうなっているかを記憶するものである。以下同様にスレッドに関する機構とメモリ領域に関する機構の同一の部分については説明を省略し、スレッドに関するものを代表して記述する。

【0054】さて、システムコールとしてユーザプログラムを実行しているあるスレッドが他のスレッドに対して制御しようとする例として、Thread_Get_Statusシステムコールを発行する場合について説明する。

【0055】このシステムコールは、スレッドIDで指定されたスレッドの状態、つまりそのスレッドのレジスタ類の情報を、このシステムコールの呼び出し元のユーザ・プログラムに通知するものである。いま、スレッド1（スレッドID=1）がプログラムA（メモリ領域ID=A）を実行中、そこでスレッド2（スレッドID=2）の情報をこのシステムコールで得ようとしたとする。C言語で記載されているプログラムAでは、次のように記されている。

```
err = Thread_Get_Status(2, &Thread_Status);
// ここで、関数Thread_Get_Statusがスレッドの情報を得るシステムコールであって、第1引数の2は、スレッド2（スレッドID=2）の情報を得ることを指示しており、第2引数の&Thread_Statusはシステムコールの結果得られたスレッド2の情報を格納するメモリ領域の先頭位置を指示している。また、このシステムコールの実行が成功したかどうかは、この関数の戻り値として変数errに入る。
```

【0056】この関数をプログラムAのスレッド1が実行すると、スレッド1はシステムコールにより実行モードがユーザレベルから特権レベルに遷移し、OSを呼び出す。特権レベルとは、OSが処理を行う専用レベルのことである。

【0057】呼び出されたOSでは、OS内のシステムコール受付部8に処理が移る。ここでは、このシステムコールを呼び出した元のユーザプログラムの中断処理を行い、要求されたシステムコールの種類と指示された引き数を受け取り、それぞれその情報を引数検出部9とシステムコールの種類判別部10に送る。

【0058】本例では、指示されたシステムコールは、Thread_Get_Statusなので、システムコールの種類判別部10で、次には、Thread_Get_Status処理部11を呼び出す。

【0059】そこでの処理は、図2に示した操作権リスト記憶部7を使って行われる。この例では操作権リスト記憶部7はスレッドに対する操作権リストを記憶するものである。この記憶部には、図2に示したようなテーブルを保持している。テーブルには、操作スレッドID、対象スレッドID、操作スレッドが存在するメモリ領域ID、スレッド操作権限、スレッド操作権変更権限の各

エリアをもっている。

【0060】同様のエリアとしては、対象スレッドが存在するメモリ領域IDを指定することも考えられるが、本例ではかかるエリアを持たないものを例として説明する。

【0061】メモリ領域に対する操作権リスト記憶部も同様であるが、テーブルの内容は若干異なる。これについての詳細は第2の実施例で説明する。

【0062】操作スレッドIDとはスレッド操作のシステムコールを実行しようとしているスレッドのスレッドIDのことで、対象スレッドIDとはこのシステムコールの引数で指定された操作対象のスレッドIDを示す。また、操作スレッドが存在するメモリ領域IDとは、この実行されようとしているシステムコールを含むプログラムの存在するメモリ領域IDのことであり、スレッド操作権限とは、どういうスレッド操作が可能を示すもので、スレッドの内部状態を読み出す権限、スレッドの内部状態を変更する権限、スレッドの実行を停止/再開する権限の3つの権限を独立に指定することができる。スレッド操作権変更権限は、スレッド操作権限の記入されたフィールドの情報を変更する権限のことであり、この権限が指定されている場合、その列に書かれているスレッド操作権限のフィールドの情報が変更できるだけでなく、その列に書かれている対象スレッドIDに関する全てのスレッド操作権限フィールドの情報を変更することができる。

【0063】たとえば図の例では、スレッド1（スレッドID=1）が発したシステムコールでは、スレッド2（スレッドID=2）の内部状態の参照と変更は可能だが、スレッド2の実行の停止/再開およびスレッド2の操作権限の変更はできないことを示している。また、スレッド1は、スレッド3（スレッドID=3）に対しては実行の停止/再開はできるがそれ以外の操作はできない。さらに、スレッド1は、スレッド2とスレッド3以外のスレッドに対しては、何の操作もできないことを示している。また、スレッド100（スレッドID=100）はスレッド2の内部状態の読み出しおよび操作権限の変更ができることを示している。

【0064】スレッド1が対象スレッドの場合の例で、テーブル中の操作スレッドが存在するメモリ領域IDの部分にa11という情報が保持されているが、これは、このシステムコールがどのユーザプログラムから行われても同等で、システムコールを実行するスレッドIDのみに実行権限が依存することを示している。

【0065】メモリ領域IDの欄がa11でないときは、そのIDのメモリ領域に存在するプログラムからシステムコールが実行された時の操作権限を規定する。たとえば、スレッド3が、プログラムA（メモリ領域ID=A）の実行中、スレッド1に対しては、すべての操作が可能であるが、それ以外のときは、操作不可である。

つまり、スレッド3がプログラムB（メモリ領域ID=B）の実行中、スレッド1に対して同様なシステムコールを実行しても操作できないし、スレッド5（スレッドID=5）がプログラムAにおいてスレッド1に対して同じシステムコールを実行しても操作できない。すなわち、このテーブルに記載のないスレッド操作は一切できない。

【0066】図2の例では、スレッド操作権限変更権限のフィールドは1つだけであるが、前記3つの権限ごとに操作権の変更権限を別々に指定できるようにしても良い。

【0067】また、スレッド操作権限には、本前の例で挙げた3つの権限の他にもあってよい。例えば、操作をするユーザのユーザID、該当フィールドを時間制限するための有効期間開始時刻、有効期間終了時刻などが考えられる。その場合にも、それら権限の変更権を設定することができる。

【0068】また、本実施例では、スレッド操作権限とスレッド操作権の変更権限とを、同一のテーブルの中で操作スレッドID、対象スレッドID、操作スレッドが存在するメモリ領域IDを共有して表現しているが、スレッド操作権限と、スレッド操作権の変更権限とを別々のテーブルに記述し、それぞれについて操作スレッドID、対象スレッドID、操作スレッドが存在するメモリ領域IDを指定しても良い。

【0069】次に、Thread_Get_Status処理部11の処理の流れを図3に示す。

【0070】まず、ステップS1にて、このシステムコールを実行した操作スレッドID、メモリ領域IDをそれぞれスレッドID記憶部6とメモリ領域ID記憶部5から情報を得る。上述したように、現在実行中のスレッドIDは、実行状態切替部4でセットされ、現在実行中のメモリ領域IDは、プロセッサの内部にあるPCの情報からセットされている。さらに、システムコールの第一引数から対象スレッドID検出部13が対象スレッドIDを得ている。第二引数からは、スレッド情報格納位置検出部14が、このシステムコールの実行結果を格納するユーザ・プログラムのメモリ位置を検出する。

【0071】次にステップS2にて、以上の3つの情報をもとに操作権記憶部71の表を引き、該当エントリをサーチし、スレッド操作権限を調べる。もし、見つければ、ステップS3に進み、なければステップS8を行うエラー処理部15へ進む。

【0072】サーチは、テーブルの上から順に比較を行い、最初にマッチしたエントリの情報を次のステップ以降で利用する。テーブルのフィールドのall部分は、すべてにマッチすることを意味する。もし、テーブルの最後のエントリまで比較し、マッチしなかったらサーチの失敗となる。

【0073】ステップS3では、見つかったエントリの

スレッド操作権限を表す部分のうち、スレッド内部状態読出権限に該当する部分の情報を読み出し、許可可否かを判断する。この判定は操作権判定部12で行う。もし判定で許可されればステップS4へ進み、不許可になればステップS8のエラー処理へ移る。

【0074】次のステップS4では、対象スレッドに対して、許可されたスレッドの内部状態を読み出すためのスレッド管理テーブルの参照操作を実行する。この処理はスレッド状態参照実行部16にて行う。ここでは、該当スレッドの内部状態であるスレッド情報を、スレッド情報格納位置検出部14で指示されたメモリ中に格納する。

【0075】ステップS5にて、ステップS4での処理結果を判定する。処理が成功するとステップS6にて、成功を示すコードをシステムコール終了部17へ伝える。失敗するとステップS7にて、失敗を示すコードをシステムコール終了部17へ伝える。

【0076】ステップS8では、エラーコードをシステムコール終了部17へ伝える。

【0077】Thread_Get_Status処理部11での処理を終えると、最後に、システムコール終了部17に処理が進む。このシステムコールを実行したユーザプログラムに返す返値をシステムコールの返値として戻るように処理を行い、中断していた呼出元のユーザプログラムの実行ができるように復帰処理を行い、実行レベルを特権レベルからユーザレベルへ戻す処理を行う。

【0078】次に、操作権記憶部71に記入されたスレッド操作権限を変更するシステムシステムコールについて説明する。

【0079】Thread_Add_Acl(指定操作スレッドID、指定対象スレッドID、指定メモリ領域ID、追加する指定操作権限)なるシステムコールは、上記で説明したスレッド操作権限を、新たに追加するものである。

【0080】いま、スレッド100（スレッドID=100）がスレッド1（スレッドID=1）に対し、スレッド2（スレッドID=2）の実行の停止/再開権限を付与しようとしたとする。C言語で記載されているプログラムCでは、次のように記されている。

【0081】err = Thread_Add_Acl(1, 2, all, THREAD_SUSPEND);ここで、関数Thread_Add_Aclは、スレッドに対する操作権を追加するためのシステムコールであって、第1引数の1は、指定した操作スレッド番号（対象スレッドID）が1であること、すなわちスレッドID=1に対して権限を与えることを指示している。第2引数の2は、指定した対象スレッド番号（指定対象スレッドID）が2であることを、すなわちスレッドID=2を操作する権限であることを指示している。第3引数のallは、指定した操作スレッドが存在するメモリ領域ID（指定メモリ領域ID）がallであること、すなわち操作スレッドが存在するメモリ領域IDに

よって与える権限を制限することはしないということを指示している。第4引数のTHREAD_SUSPENDは、これら3つの指定した組み合わせに対して、付加する操作権限(指定操作権限)を指示している。この例では、スレッドの実行の停止/再開権限を付加することを意味している。第4引数には複数種類の権限を同時に指定することもできる。

【0082】この関数をプログラムCのスレッド100が実行すると、スレッド100はシステムコールによりプロセッサの実行モードをユーザレベルから特権レベルに遷移させ、OSを呼び出す。特権レベルとは、OSが処理を行う専用レベルのことである。

【0083】呼び出されたOSでは、OS内のシステムコール受付部8に処理が移る。ここでは、このシステムコールを呼び出した元のユーザプログラムの中断処理を行い、要求されたシステムコールの種類と指示された引数を受け取り、それぞれその情報を引数検出部9とシステムコールの種類判別部10に送る。

【0084】本例では、指示されたシステムコールは、Thread_Add_Aclなので、システムコールの種類判別部10で、Thread_Add_Acl処理部21を呼び出す。Thread_Add_Acl処理部21は図1におけるThread_Get_Status処理部11を図4のように置き換えたものである。なお、Thread_Add_Acl処理部21とThread_Get_Status処理部11は本来一体のブロック図として示すべきものであるが、図が煩雑になるため説明の便宜をはかり図1および図4の別々の図としてある。

【0085】そこでの処理は、図2に示したスレッド操作権限リスト記憶部7を使って行われる。その内容および意味は前に説明した通りである。

【0086】次に、図5にThread_Add_Acl処理部21の処理の流れを示す。

【0087】まず、ステップS11にて、このシステムコールを実行した実行スレッドID、メモリ領域IDをそれぞれスレッドID記憶部6とメモリ領域ID記憶部5から情報を得る。さらに、システムコールの第2引数から指定対象スレッドID検出部23が、このシステムコールによってスレッド操作権限の変更を行う対象としたスレッドのIDを得る。

【0088】次にステップS12にて、以上の3つの情報をもとにスレッド操作権限リスト記憶部7のテーブルを引き、該当エントリをサーチする。もし、見つければ、ステップS13に進み、なければステップS19を行うエラー処理部15へ進む。サーチは、テーブルの上から順に比較を行い、最初にマッチしたエントリの情報を次のステップ以降で利用する。テーブルのフィールドのall部分は、全てにマッチすることを意味する。もし、テーブルの最後のエントリまで比較し、マッチしなかったらサーチの失敗となる。

【0089】ステップS13では、見つかったエントリ

のスレッド操作権限変更権限に該当する部分の情報を読み出し、許可可否かを判断する。この判定は操作権限変更権限判定部26で行う。もし判定で許可されればステップS14へ進み、不許可になればステップS19のエラー処理へ移る。

【0090】ステップS14では、システムコールの第1引数から、指定対象スレッドID検出部23が指定した対象スレッドIDを得る。また、システムコールの第3引数から、指定メモリ領域ID検出部24が指定した操作スレッドが存在するメモリ領域IDを得る。さらに、システム・コールの第4引数から、指定操作権限検出部25が指定した追加すべき操作権限を得る。

【0091】ステップS15では、ステップS11およびステップS14で得られた対象スレッドID、指定対象スレッドID、指定メモリ領域ID、指定操作権限をもとに操作権限リスト変更部27が操作権限リスト記憶部7を書き換える。つまり、まず操作権限リスト記憶部7のテーブルを順にサーチし、対象スレッドIDと指定対象スレッドIDと指定メモリ領域IDの3つが全て一致する列があれば、その列のスレッド操作権限のフィールドを書き換え、もしなければ、指定した3つのIDを示す列を新たに操作権限リスト記憶部7のテーブルに追加する。この例では、指定した3つのIDと一致する列が最初に見付かるので、その列のスレッド操作権限のフィールドにあるスレッドの実行の停止/再開権限を×から○に変更する。

【0092】ステップS16にて、ステップS15での処理結果を判定する。処理が成功するとステップS17にて、成功を示すコードをシステムコール終了部17へ伝える。失敗するとステップS18にて、失敗を示すコードをシステムコール終了部17へ伝える。ステップS19では、エラーコードをシステムコール終了部17へ伝える。

【0093】Thread_Add_Acl実行部21での処理を終えると、最後にシステムコール終了部17に処理が進む。このシステムコールを実行したユーザプログラムに返す返値をシステムコールの返値として戻るように処理を行い、中断していた呼出元のユーザプログラムの実行ができるように復帰処理を行い、実行レベルを特権レベルからユーザレベルへ戻す処理を行う。

【0094】本例のようなThread_Add_Aclシステム・コールの実行が成功した結果、図2で示された操作権限リスト記憶部7のデータは図6のように変更される。

【0095】本例では操作権限リスト変更部27において、書換を必要とする該当フィールドを書き換えるだけで、Thread_Add_Aclシステム・コールからの要求を実行することができたが、Thread_Add_Aclシステム・コールに与えられた引数の内容によっては、該当フィールドを書き換えるだけでは表現できず、操作権限リスト記憶部7のデータを増やさなくてはならない。

【0096】例えば、操作権リスト記憶部7のデータが先の図2に示したとおりであるとき、Thread_Add_Acl(3, 2, A, THREAD_SUSPEND*THREAD_GET_STATUS)を実行した場合を考える。

【0097】ここで、第1引数の3は、指定した操作スレッド番号(対象スレッドID)が3であること、すなわちスレッドID=3に対して権限を与えることを指示しており、第2引数の2は、指定した対象スレッド番号(指定対象スレッドID)が2であること、すなわちスレッドID=2を操作する権限であることを指示しており、第3引数のAは、指定メモリ領域IDがAであること、すなわち、このシステムコールの実行により変更される権限は、プログラムA(メモリ領域ID=A)を実行中にのみ有効であることを指示しており、第4引数のTHREAD_SUSPENDは、これら3つの指定した組み合わせに対して、スレッドの実行の停止/再開権限を付加すること、そしてTHREAD_GET_STATUSはスレッドの内部状態読出権限を付加することを指示している。

【0098】このシステム・コールが発せられると、操作権リスト変更部27が、操作権リスト記憶部7のテーブルを順にサーチし、対象スレッドIDと指定対象スレッドIDと指定メモリ領域IDの3つが全て一致する列をさがすが、この場合そのような列は存在しないので、このシステム・コールは、操作権リスト記憶部7の持つテーブルの一部のフィールドを書き換えるだけでは実行できない。指定した3つのIDを示す列を新たに操作権リスト記憶部7のテーブルに追加する。このような手順を踏むことにより、このシステム・コールの実行が終了すると、図2で示された操作権リスト記憶部7のデータは図7のように変更され、操作権リスト記憶部7の列の長さが1つ長くなっている。

【0099】以上の説明においては、最初に操作権変更権限をどこに指定するかということについては記述されていない。一般にはOSの起動時に、一部の特権的なスレッドに対してこの権限を与え、以下階層的に権限を増やしてゆくことにより実現することが可能である。

【0100】しかし、スレッドの作成時に必ず操作権変更権を指定しなければならないのは、プログラミングの手間を増やすおそれがあるので、メモリ領域またはスレッドの作成時には、あらかじめ定められたデフォルトの操作権変更権が設定されるように構成しても良い。デフォルトを何にするかについては各種考えられるが、最も簡単には、そのメモリ領域またはスレッドを起動したスレッドに対して操作権変更権を与えることが考えられる。この方法は、デフォルトとして従来OSにおけるオーナーの概念を用いたことになり、さらに本実施例で説明した方法により、本発明の操作権管理装置はより柔軟な操作権管理を実現するものである。

【0101】また、先の例ではThread_Add_Aclの第4引数に指定することのできる指定操作権限に制約はな

いものとして記述したので、第4引数にTHREAD_CHANGE_ACLを指定することができる。すなわち、スレッド操作権変更権限を変更させることもできる。また、以上説明したThread_Add_Aclと同様に、Thread_Del_Aclなるシステム・コールを用いて、例えば

Thread_Del_Acl(3, 1, A, THREAD_CHANGE_ACL)

は、スレッドが、メモリ領域ID=Aのメモリ領域から、スレッドID=1のスレッドのスレッド操作権限を取り除くことを意味している。この場合、永遠に誰にも操作権変更権限を変更することのできないようなスレッドを作ってしまうおそれがあるが、そういった変更が行われることが問題となるオペレーティングシステムにおいては、操作権変更権記憶部の情報を変更する際に、操作権記憶部の情報を変更することが不可能になるような変更を禁止するようにしても良い。例えば、先にあげたThread_Del_Acl(3, 1, A, THREAD_CHANGE_ACL)

を実行すると、もはやスレッドID=1のスレッドに対し、スレッド操作権限を変更することのできるスレッドが1つもなくなる。このようなシステム・コールが実行されると、操作権リスト変更部27はエラーを検出したことになり、失敗を示すコードをシステムコール終了部17へ伝え、操作権リスト記憶部のデータは変更されない。

【0102】また、操作権変更権を変更する権利を別個のものとして扱い、操作権リスト記憶部7に操作権変更権管理権限のフィールド(操作権変更権管理権限記憶部)を用意し、より確実な操作権管理を行っても良い。

【0103】さらには、操作権変更権記憶部の情報を変更することが不可能になるような、操作権変更権管理権限記憶部の情報の変更を禁止するようにしても良い。

【0104】なお、上記「スレッドID」を「ユーザID」に置き換えて構成することも可能である。この場合、図1におけるスレッドID記憶部6を、ユーザID記憶部と置き換える修正を施せば良い。

【0105】また、前述した「操作スレッドが存在するメモリ領域ID」を「操作スレッドが実行しているプログラムID」に置き換えて構成することも可能である。この場合、図1におけるメモリ領域ID記憶部5は、プログラムID記憶部と置き換える修正を施せば良い。

【0106】(第2の実施例)次に、本発明の第2の実施例について説明する。

【0107】本発明の操作権管理装置は、メモリ領域やスレッドに対する操作権管理を、例えばオーナーといった固定的な概念により行うのではなく、操作権変更権を操作権リスト自体に持たせ、自由に削除、追加させることによって柔軟に行うことを目的としている。ここでは、この機構を生かした例として、複数人による共同作業のアプリケーションが動作する場合に、操作権管理が柔軟に行われることを示す実施例について述べる。

【0108】ユーザーAは、計算機を使った電子会議を

開催すべく、新たにスレッド10（スレッドID=10）を作成した。より具体的には、ユーザーAの動かししていたログイン・シェルであるスレッド11がスレッド10を作成した。このとき、操作権リスト記憶部7の該当部分のデータは図8のようになっている。以下図には操作権リスト記憶部7のテーブルのうち、本実施例を説明するのに必要な部分のみを記述し、他の部分については記述を省略する。

【0109】スレッド11（スレッドID=11）はスレッド10（スレッドID=10）の内部状態読み出しおよびスレッド操作権の変更ができる。しかし、内部状態を変更したり、実行の停止／再開を行うことはできない。つまり、既に動き始めたスレッド10（スレッドID=10）は、それを作成したスレッド11（スレッドID=11）によって誤って実行を妨害されることはない。これらの初期状態はスレッドを作成するシステムコールの引数として指定するものとする。

【0110】作成されたスレッド10は電子会議のプログラムを実行し、ユーザーAとのインターフェースを司る。また、スレッド10は電子会議をサポートするためのメモリ領域として、メモリ領域30（メモリ領域ID=30）を作成する。

【0111】メモリ領域30に関する操作権の管理には、スレッドの操作権の管理と同様に操作権リスト記憶部7を用いる。ただし、スレッドの操作権管理の説明で述べた対象スレッドID、スレッド操作権限、スレッドの内部状態読出権限、スレッドの内部状態変更権限、スレッドの実行の停止／再開権限、スレッド操作権変更権限は、メモリ領域の操作権管理においてはそれぞれ対象メモリ領域ID、メモリ領域操作権限、メモリ領域の内部データ読出権限、メモリ領域の内部データ変更権限、メモリ領域にあるプログラムの実行権限、メモリ領域操作権変更権限、となる。

【0112】図9に示した操作権リスト記憶部は、スレッド10がメモリ領域30の内部データの読み出しおよび変更、操作権変更ができることを意味している。

【0113】次にユーザーBおよびユーザーCが電子会議への参加を表明し、スレッド10は新たにユーザーBおよびユーザーCのためのスレッド12（スレッドID=12）およびスレッド13（スレッドID=13）を作成した。その結果、図8の操作権リスト記憶部7のデータは図10のように追加される。また電子会議をサポートするためのメモリ領域であるメモリ領域30は、スレッド12およびスレッド13からも使われるので、図9の操作権リスト記憶部7のデータは図11のように追加される。

【0114】つまり、スレッド12およびスレッド13はスレッド10によりコントロールされており、スレッドに関するあらゆる操作ができるが、スレッド12およびスレッド13は他のスレッドのコントロールをするこ

とはできない。これにより、スレッド10が先導して電子会議の進行をプログラムされた通りに実行することができる。

【0115】共通のメモリ領域として使われるメモリ領域30は、スレッド10、スレッド12、スレッド13のいずれからもデータの読み出し、変更ができるが、その領域が誰から（どのスレッドから）どのような操作ができるかということを変更することができるのはスレッド10だけである。

【0116】このような仕組みで、複雑な依存関係のある電子会議のようなコミュニケーション・プログラムもOSの操作権管理機構を用いて確実に記述することができる。

【0117】次に、会議を先導していたユーザーAのスレッド10が、途中で会議を抜け、会議はそのまま進行している場合を考える。

【0118】単純にスレッド10が自分自身の実行を終了させたのでは、メモリ領域30の操作権を変更することのできるスレッドがいなくなり、会議を正しく進行させることができなくなる。このような場合には、ユーザーAは一旦会議の進行権を他の人（例えばユーザーB）に譲らなければならない。

【0119】まず、メモリ領域30の操作権をスレッド12（ユーザーBのスレッド、スレッドID=12）に設定する。そして、スレッド13の管理権をスレッド12に譲り、そのあとで実行を終了する。それ以外のもので、操作スレッドIDが10であるものは、スレッド10の終了と同時に自動的に消滅させる。

【0120】これによって、図10および図11の操作権リスト記憶部はそれぞれ図12および図13のようになる。すなわち、メモリ領域30へのアクセスはスレッド12とスレッド13の両方からできるが、スレッドの管理自体の権利はスレッド12が継承する。注意すべき点は、スレッド10の実行が終了したために、ユーザーAのlogin shellであるスレッド11からは、電子会議を実行するいずれのプログラムに対しても何の権利もなくなっていることである。

【0121】ここに示した例は電子会議プログラムを実現する一方法を示したものであり、必ずしもこのような手順で操作権を譲渡してゆかなければプログラムが記述できないということは意味していない。

【0122】このようにして、スレッドおよびメモリ領域への操作権を管理しつつ、管理者の変更を実現することができる。

【0123】本発明の操作権管理装置を使えば、「会議に参加する」あるいは「会議から退室する」という、本実施例で具体的に例をあげて説明したような場合における、曖昧な操作権管理も、アプリケーションで逐一記述することなく、OSのサポートにより柔軟かつ確実に実現することができる。

【0124】また、本発明は上述した実施例に限定されるものではなく、その要旨を逸脱しない範囲で、種々変形して実施することができる。

【0125】

【発明の効果】本発明の操作権管理装置によれば、スレッドまたはメモリ領域に対する操作を保護するための操作権管理を柔軟かつ確実に行うことができる。

【図面の簡単な説明】

【図1】 Thread_Get_Statusを説明するための構成図

【図2】図1の操作権リスト記憶部7の内部構成例を示す図

【図3】本実施例に係わる操作権管理装置において操作権を変更する動作を示すフローチャート

【図4】 Thread_Add_Acl を説明するための構成図

【図5】本実施例に係わる操作権管理装置において操作権変更権を変更する動作を示すフロー図

【図6】図4の操作権リスト記憶部7の内部構成例を示す第1の図

【図7】図4の操作権リスト記憶部7の内部構成例を示す第2の図

【図8】電子会議開催時のスレッド操作権リストを示す図

【図9】電子会議開催直後のメモリ領域操作権リストを

示す図

【図10】電子会議に新たなメンバが参加した際のスレッド操作権リストを示す図

【図11】電子会議に新たなメンバが参加した際のメモリ領域操作権リストを示す図

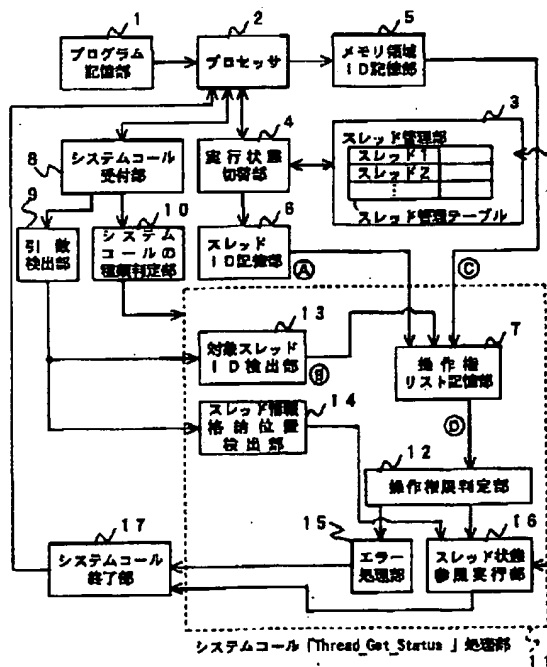
【図12】ユーザーAが電子会議の先導権をユーザーBに委譲した際のスレッド操作権リストを示す図

【図13】ユーザーAが電子会議の先導権をユーザーBに委譲した際のメモリ領域操作権リストを示す図

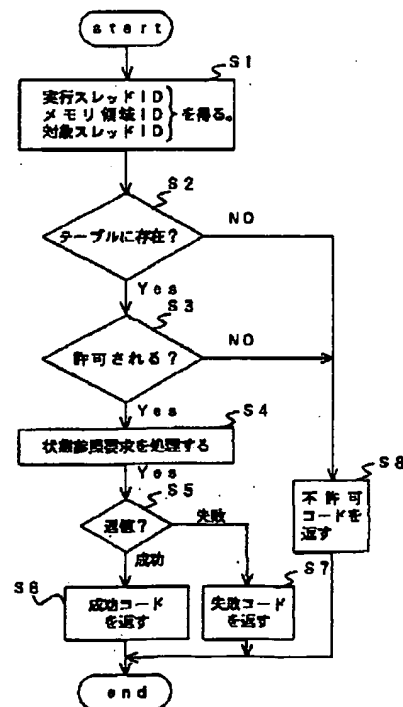
【符号の説明】

1…プログラム記憶部、2…プロセッサ、3…スレッド管理部、4…実行状態切替部、5…メモリ領域ID記憶部、6…スレッドID記憶部、7…操作権リスト記憶部、8…システムコール受付部、9…引数検出部、10…システムコールの種類判定部、11、21…システムコール実行部、12…操作権限判定部、13…対象スレッドID検出部、14…スレッド情報格納位置検出部、15…エラー処理部、16…スレッド状態参照実行部、17…システムコール終了部、22…指定操作スレッドID検出部、23…指定対象スレッドID検出部、24…指定メモリ領域ID検出部、25…指定操作権限検出部、26…操作権変更権判定部、27…操作権リスト変更部

【図1】

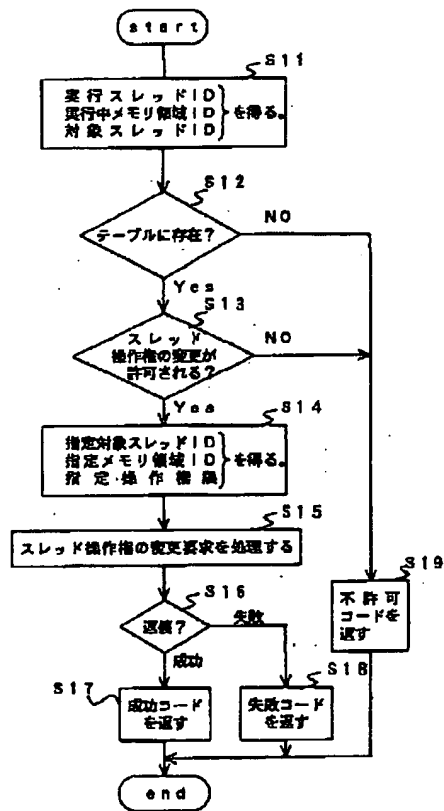


【図3】



操作権記憶部 T 1 操作権変更権記憶部 T 2

【図4】



操作 スレッド ID	対象 メモリ領域 ID	操作スレッドが 存在するメモリ 領域 ID	メモリ領域操作権限			メモリ領域 操作権変更権限
			メモリ領域の 内部データ 読み権限	メモリ領域の 内部データ 書き込み権限	メモリ領域の プログラム 実行権限	
10	30	all	○	○	×	○

【図6】

操作スレッドID	対象スレッドID	操作スレッドが存在するメモリ領域ID	スレッド操作権限			スレッド操作権変更権限
			スレッドの内部状態読出権限	スレッドの内部状態変更権限	スレッドの実行の停止/再開権限	
1	2	all	○	○	○	×
1	3	all	×	×	○	×
2	1	all	×	×	○	×
3	1	A	○	○	○	○
100	2	all	○	×	×	○

【図7】

操作スレッドID	対象スレッドID	操作スレッドが存在するメモリ領域ID	スレッド操作権限			スレッド操作権変更権限
			スレッドの内部状態読出権限	スレッドの内部状態変更権限	スレッドの実行の停止/再開権限	
1	2	all	○	○	×	×
1	3	all	×	×	○	×
2	1	all	×	×	○	×
3	1	A	○	○	○	○
100	2	all	○	×	×	○
3	2	A	○	×	○	×

【図8】

操作スレッドID	対象スレッドID	操作スレッドが存在するメモリ領域ID	スレッド操作権限			スレッド操作権変更権限
			スレッドの内部状態読出権限	スレッドの内部状態変更権限	スレッドの実行の停止/再開権限	
11	10	all	○	×	×	○

【図10】

操作スレッドID	対象スレッドID	操作スレッドが存在するメモリ領域ID	スレッド操作権限			スレッド操作権変更権限
			スレッドの内部状態読出権限	スレッドの内部状態変更権限	スレッドの実行の停止/再開権限	
11	10	all	○	×	×	○
10	12	all	○	○	○	○
10	13	all	○	○	○	○

【図11】

操作 スレッド ID	対象 メモリ領域 ID	操作スレッドが 存在するメモリ 領域ID	メモリ領域操作権限			メモリ領域 操作権限変更権限
			メモリ領域の 内部データ 読み出し権限	メモリ領域の 内部データ 書き込み権限	メモリ領域の プログラム 実行権限	
10	30	all	○	○	×	○
12	30	all	○	○	×	×
13	30	all	○	○	×	×

【図12】

操作 スレッド ID	対象 スレッド ID	操作スレッドが 存在するメモリ 領域ID	スレッド操作権限			スレッド操作 権限変更権限
			スレッドの内部 状態読み出し権限	スレッドの内部 状態書き込み権限	スレッドの実行の 停止/再開権限	
12	13	all	○	○	○	○

【図13】

操作 スレッド ID	対象 メモリ領域 ID	操作スレッドが 存在するメモリ 領域ID	メモリ領域操作権限			メモリ領域 操作権限変更権限
			メモリ領域の 内部データ 読み出し権限	メモリ領域の 内部データ 書き込み権限	メモリ領域の プログラム 実行権限	
12	30	all	○	○	×	○
13	30	all	○	○	×	×

フロントページの続き

(72)発明者 申 承昊

神奈川県川崎市幸区小向東芝町1番地 株
式会社東芝研究開発センター内

(72)発明者 吉田 英樹

神奈川県川崎市幸区小向東芝町1番地 株
式会社東芝研究開発センター内